Enhanced medial-axis-based block-structured meshing in 2-D

Enhanced medial-axis-based block-structured meshing in 2-D

Harold J. Fogg, Cecil G. Armstrong, Trevor T. Robinson*

School of Mechanical and Aerospace Engineering, Queen's University Belfast, BT9 5AH, N. Ireland.

Abstract

New techniques are presented for using the medial axis to generate decompositions on which high quality block-structured meshes with well-placed mesh singularities can be generated. Established medial-axis-based meshing algorithms are effective for some geometries, but in general, they do not produce the most favourable decompositions, particularly when there are geometric concavities. This new approach uses both the topological and geometric information in the medial axis to establish a valid and effective arrangement of mesh singularities for any 2-D surface. It deals with concavities effectively and finds solutions that are most appropriate to the geometric shapes. Resulting meshes are shown for a number of example models.

Keywords: medial axis, quad mesh, multiblock decomposition, block-structured, mesh singularities, cross-field.

1. Introduction

An assessment of the relative performances of multiblock structured meshes of turbomachinery that were generated by medial-axis-based methods, Cartesian grid methods and by manual interactive work was made in a recent paper [1]. Adjoint error analyses in CFD simulations showed that the medial-axis-based methods generated higher quality meshes and this can be attributed to their boundary-sympathetic block configurations. But despite their relative effectivenesses, there are still some obvious improvements to be made, such as more natural treatment of concavities and responsiveness to

*Correspond to t.robinson@qub.ac.uk

geometric shapes. Addressing these limitations by employing the neglected geometric information in the medial axis is the objective of this work.

1.1. Medial axis definition

For a 2-D surface, the *medial axis* is defined as the set of points where each point, $\hat{\mathbf{p}}$, has a centred inscribed circle, U, that touches the boundary entities at more than a single point. Distinct parts of the medial axis are characterised by the number of touching points of $U(\hat{\mathbf{p}})$.

Medial edges connect subsets of the points with two touching points and they usually make up the majority of the medial axis. The radius of $U(\hat{\mathbf{p}})$ is called the *medial radius*, r_m, and the subtended angle between the radii of $U(\hat{\mathbf{p}})$ to the touching points is called the *medial angle*, θ_m. A special type of medial edge feature that frequently occurs is termination at a convex corner where r_m shrinks to zero. The corresponding medial edge is called a *flare* or *flange*. These are all illustrated in Fig. 1.

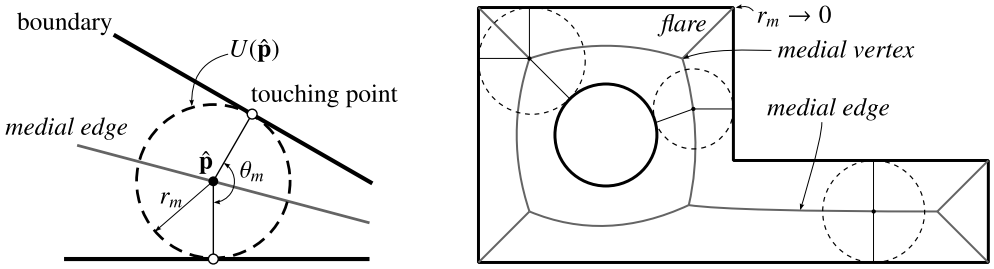


Figure 1: Medial edge features and terminology (left) and medial axis for simple surface (right).

Medial vertices represent the meeting points of medial edges as $U(\hat{\mathbf{p}})$ transitions from one boundary pair to another. In standard cases three touching points are involved. In special cases a medial vertex is equidistant to more than three touching points. There are two other special cases, the first being a medial vertex at the centre of a circular arc with $U(\hat{\mathbf{p}})$ in finite contact. The other is a curvature contact scenario, such as near the end of the major principal axis of an ellipse. There, the curvature of $U(\hat{\mathbf{p}})$ equals that of the boundary at the touching point, which results in a single touching point belonging to a medial vertex with θ_m equal to zero (cf. Fig. 10).

1.2. Review of established medial-axis-based decomposition algorithms

Quad meshing by medial axis decomposition was first conceived by Nackman and Srinivasan [2]. They noticed that the medial axis presented a well-

defined natural decomposition of the surface and used it without alterations to split the surface into blocks. Some additional rules for merging the produced blocks were devised to improve the suitability of the decomposition for meshing.

A better method was proposed by Tam and Armstrong [3] that produces higher quality decompositions. They developed a strategy for using a Constrained Delaunay Triangulation (CDT) of a set of boundary points, whose circumcenters approximate the medial axis to an acceptable accuracy, to assemble 'shape molecules' characterising a mesh topology. By inserting cuts between the medial vertices the shape molecules are reduced to 'shape atoms' which are topological 4-, 5- and 6-sided polygons. Each of these are meshable by midpoint subdivision [4] and an integer programming method can be used to find allowable division numbers on the subblock edges [5, 6]. An augmented algorithm with this approach has been implemented in the commercial software of Abaqus [7], offering automatic high-quality meshing capability.

More recently, Rigby [8] proposed an algorithm called 'TopMaker'. It consists of a few simple rules for creating suitable cuts at medial vertices and along the medial axis and it is surprisingly effective in generating pleasing decompositions for a certain class of geometries.

Additional related methods include [9] which is less general and has rules specifically tailored for managing circular holes. The 'd-MAT' method used by Xia and Tucker [10, 1] has similar rules to the TopMaker method and it produces comparable decompositions.

1.3. Other quad mesh generation methods

Mapped meshing [11] is the most long-established method for generating quad meshes. The standard method involves solving for a boundary-fitted curvilinear coordinate system on the surface. The range of surfaces for which it is effective is limited to near rectangular simply connected surfaces and regular annuli. Mapped meshing with other simple mesh topologies has also been developed for triangular and polygon surfaces [12]. Submapping [13] extends the range of applicable surfaces to include those with 'blocky' profiles, not necessarily with convex corners but which allow logical mappings of the boundaries to a Cartesian coordinate system. The decomposition strategy for quad meshing, such as the aforementioned medial-axis based methods, is to subdivide complex domains into quadrilateral, polygon and 'blocky' subregions on which mapped meshing methods can be used.

The advancing front or paving method [14] involves building layers of elements in stages, working inwards from the boundaries. When layers overlap the element connectivities are readjusted to produce a valid mesh. This often results in the creation of many irregular valence nodes, or singularities, along the collision zones of layers, which is typically around the medial axis. Unconstrained paving [15] achieves more regular mesh patterns by identifying common configurations in the local geometry and topology before resolving the mesh connectivities in the collision zones, although there are some heuristic parameters that must be fine tuned for each geometry.

Traditional Cartesian grid methods [16] comprise overlaying regular grids on the geometry. Elements that intersect with the boundary are removed and then the remaining outer elements are adapted and added to generate a boundary conformal mesh. While this approach generates meshes that are predominately structured, the mesh on the boundaries is inevitably irregular and of poor quality. Much better meshes can be produced by new methods which first seek a transformed representation of the surface where every edge follows a Cartesian coordinate system before applying a Cartesian grid [17] . This is similar in principal to the submapping method. The grid transformed back to the original surface has regular connectivities and generally has good element qualities . However, the transformation may unavoidably have undesirable singularities on the boundary which cause poor mesh quality locally. A simple example of a geometry where the problem arises is a triangle. Ideally, the necessary singularities should occur in the interior to keep boundary element quality high and to minimise the overall mesh distortion.

The newest most advanced algorithms for quad mesh generation include Morse-Smale complex methods [18] and cross-field methods [19, 20, 21, 22]. Both approaches solve scalar and or vector fields on the surface from which quadrangular charts are extracted and meshed by parametrisation methods. Impressive high-quality quad meshes with target size and direction sensitivity can be produced by these methods. Their downside is that they are complex, quite expensive and require a suitable triangulation of the surface.

A method with some similarities to the presented method is LayTracks [23]. Its strategy for mesh generation is fundamentally different to decomposition based methods – it uses the medial axis to generate a mesh directly as opposed to of using it to infer mesh properties and with that information create a decomposition. The algorithm creates regularly spaced thin strips between medial radii pairs from medial axis points to touching points. These strips are then divided up into quadrilateral pieces. In common with the presented

method, the mesh singularities are pushed out to the interior and high quality elements are produced at the boundaries. In contrast to the presented method, at concavities the mesh rows curl around the sharp tips and elements tend to degenerate to triangles, similar to the TopMaker method.

1.3.1. Strengths and weaknesses of TopMaker and Tam and Armstrong (T&A) methods

In comparison to other widely used general quad meshing algorithms, such as paving or Cartesian grid methods, the TopMaker and T&A algorithms can usually generate superior block-structured meshes; they have fewer mesh singularities and simpler topologies than paved meshes and have higher quality elements on the boundaries than Cartesian meshes. However, they are in fact only effective for certain types of surfaces.

Neither handles concavities particularly well. The TopMaker algorithm makes the assumption that all corners are flat except those connected to flares which have one element. Using the definitions given in Sec. 3 (Fig. 6 (right)), these translate as $n_c = 2$ and $n_c = 1$ type corners respectively. Whilst this makes the decision making in the algorithm very straightforward, it doesn't always lead to serviceable decompositions. Consider the decomposition shown in Fig. 2 (top right). The mesh is forced to wrap around the trailing edges which leads to highly distorted blocks. Therefore, in general, acceptable quality decompositions are only created for geometries with corner angles $<\sim 5\pi/4$.

The T&A method performs slightly better with concavities as borne out in Fig. 2 (bottom right). In the first stage the surface is split by the best candidate CDT edges that fit most closely with the assigned mesh patterns at the corners, which are chosen based on corner angles. The splitting edges for concavity removal are shown in Fig. 2 (bottom left). This strategy can at least produce $n_c = 3$ and 4 type corner patterns but it often does not lead to the best overall mesh topology. For example, the method always creates a split between concave vertices that share an inscribed circle and this results in the sub-optimal decomposition shown in Fig. 3 (centre) for the *indents* surface. But the most obvious and best quality decomposition is shown in Fig. 3 (right).

Another limitation is the almost exclusive use of topological information in the medial axis and the disregard of important embedded geometrical shape information. For example, each of the three surfaces shown in Fig. 4 (top row) are the same as far as the TopMaker and T&A algorithms are

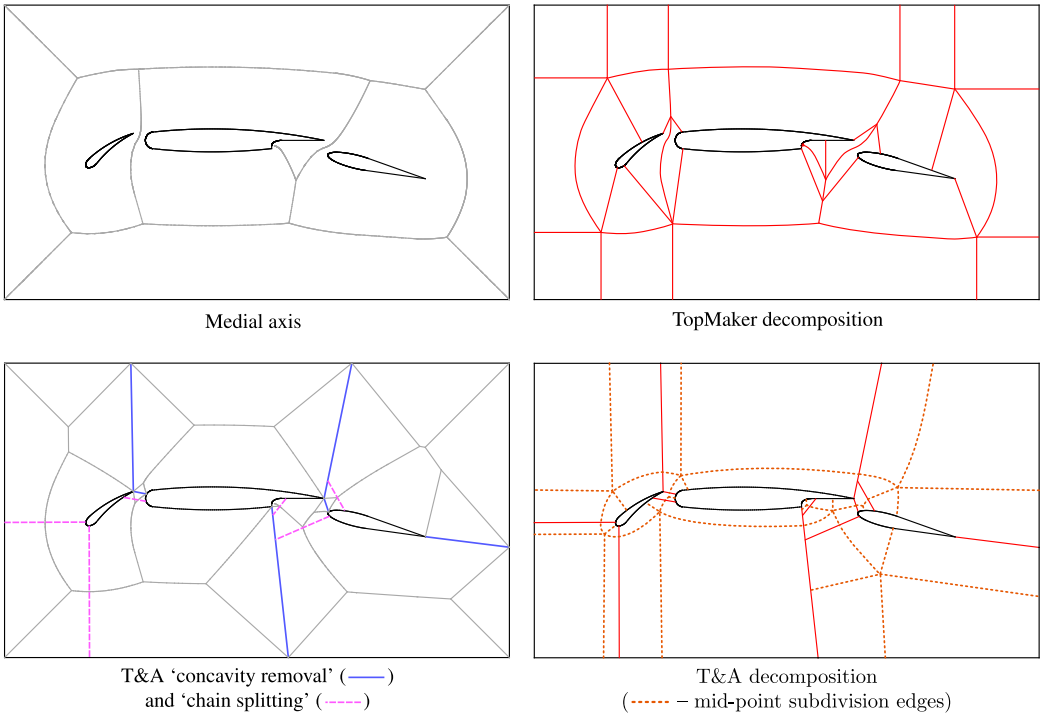


Figure 2: *Multi-element aerofoil* surface decomposed by TopMaker and T&A algorithms.

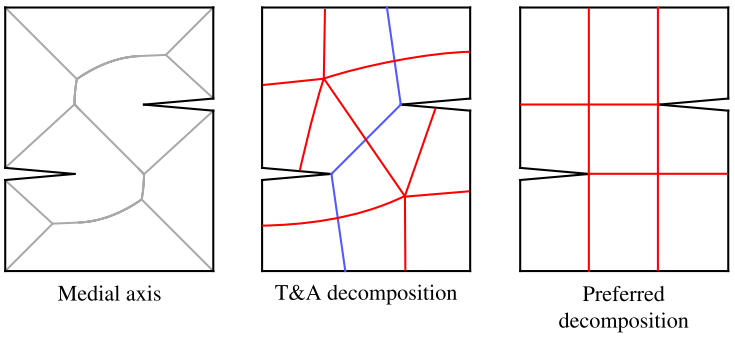


Figure 3: *Indents* surface decomposed by T&A algorithm and its preferred decomposition.

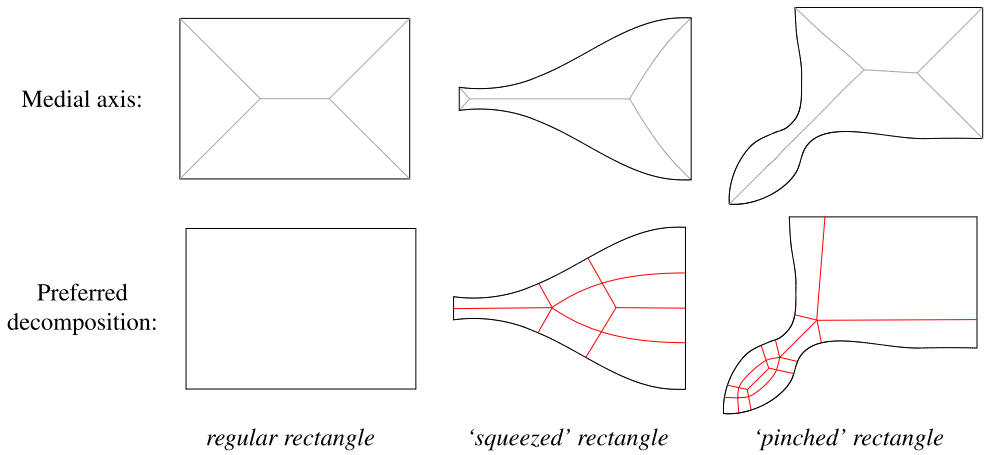


Figure 4: Three surfaces with an identical medial axis topology but quite different geometric shapes (top row) and suggested good quality decompositions (bottom row).

concerned because the medial axis topologies are identical. Thus, they are all treated as rectangular blocks for which a regular grid can be used in their interiors. This is appropriate for the first surface but a regular grid mapped to the second and third will be highly distorted. Preferably, they would be decomposed, or more specifically, the block topology solutions should contain some positive-negative singularity pairs, to give the block structures sketched in Fig. 4 (bottom row – middle and right).

1.4. Contributions

Herein, a new approach is described which combines the medial-axis-based decomposition algorithms and the frame-field decomposition approaches [24, 19, 25]. It makes use of the valuable and hitherto neglected geometric information in the medial axis, namely the medial angle, which leads to the development of a simple well-reasoned method for finding a suitable configuration of mesh singularities on the medial axis. This is essentially the critical step in a mesh generation problem, as explained briefly in the next section. It also gives a basic description of element orientation over the whole surface.

Based on this information, a new medial-axis-based decomposition method is presented. It provides a general way of handling concavities and is precisely responsive to important geometric shapes so that preferred solutions like those shown in Fig. 3 (right) and Fig. 4 (bottom row) are found automatically.

The method is applicable to all surfaces with computable medial axes, which are non-closed surfaces with boundaries. The more curved the surface is the less worthwhile the idea becomes that boundary alignment constraints should govern its mesh solution. For this reason the presented method, and indeed all medial-axis-based methods, are most suited to near-planar surfaces or small patches of curved ones.

2. Theoretical properties of quad meshes

2.1. Introduction

The new techniques are reasoned in terms of a rigorous theoretical description of unstructured quad meshes with infinitesimal properties that was described by Bunin [26]. The theory gives insight into the fundamental properties of quad meshes and explains the crucial role played by mesh singularities, which are nodes where the regular grid connectivities are disrupted.

2.2. Overview of the theory

The key results of Bunin's theory are:

- The governing equation of an orthogonal mesh is

$$\Delta_S \phi = K + \sum_{i=1}^{N} k_i \frac{\pi}{2} \delta_{\mathbf{p}_i}, \quad k_i \in \mathbb{Z} \geq -4. \tag{1}$$

  This is a Poisson equation and it describes stationary heat conduction amongst other physical phenomena. Δ_S is the Laplace-Beltrami operator where the subscript S indicates that it is measured with respect to the parametric space of the surface and not the 3-D space that the surface is embedded in. ϕ is a scalar field on the surface that controls the mesh, K is the Gaussian curvature of the surface, k_i is the type of a mesh singularity and $\delta_{\mathbf{p}_i}$ is a Dirac delta function centred at the point on the surface where the singularity occurs, $\mathbf{p}_i$.

- The type of a mesh singularity is given by an integer, k, which essentially denotes the number of additional elements above four at a node.

- The variable ϕ corresponds to $-\ln h$, where h is the edge length of an infinitesimal square element.

- The curvatures of the mesh edges are related to the variation in the ϕ-field by

$$\kappa_g = \frac{\partial \phi}{\partial e} \equiv \langle \nabla_S \phi, \mathbf{e} \rangle, \tag{2}$$

  where κ_g denotes the geodesic curvature of the edge and its intrinsic normal, $\mathbf{e}$, is defined by $\mathbf{e} = \mathbf{n} \times \mathbf{t}$ where $\mathbf{n}$ and $\mathbf{t}$ are the surface normal and the edge tangent vector respectively.

- A cross-field describing the mesh directionality (see [27, 28, 29, 24, 19]) is closely related to the ϕ-field. The total change in angle of a cross along a curve, α, (in a parallel-transport sense) is given by

$$\Delta\theta = \int_\alpha \frac{\partial \phi}{\partial e} ds. \tag{3}$$

With regard to the problem of mesh generation, a continuum description of a mesh can be obtained by solving a heat conduction type problem (by FEM for example) once a valid arrangement of mesh singularities have been specified. But finding the positions of mesh singularities is not trivial. Indeed, this theory shows that it falls into the category of Inverse Poisson problems which are known to be ill-posed and difficult to solve effectively [30].

3. Resolving mesh singularities using the medial axis

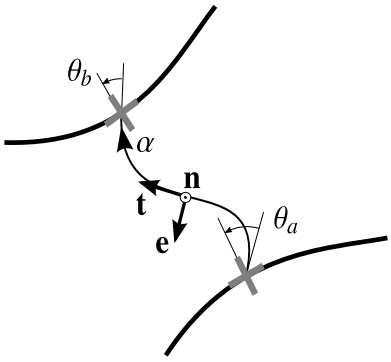
3.1. Optimum mesh-flow indicated by the medial angle

The medial axis supplies a link between nearby boundaries and provides a means to assess the geometry locally. An intuitive idea is that the relative orientations of the boundaries indicate preferred patterns of the mesh in the intermediate areas. It is convenient to use a cross-field representation of the mesh to demonstrate this.

Given two points with assigned crosses, the total change in angle of any cross-field between them is unique up to a multiple of $\pi/2$. The multiplying integer, call it n, essentially specifies the number of $\pi/2$ turns made by a cross moving between the points. With reference to Bunin's continuum theory, the total flux through a curve linking the points is given by

$$\Phi_\alpha \equiv \int_\alpha \frac{\partial \phi}{\partial e}\, ds = \int_\alpha \kappa_g\, ds + \theta_b - \theta_a - n\frac{\pi}{2}, \tag{4}$$

with the symbol definitions as indicated in Fig. 5 (left). A smooth regular



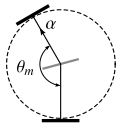
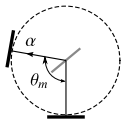
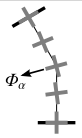
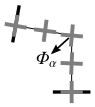
θ_m range	$[\pi, 3\pi/4)$	$[3\pi/4, \pi/4)$	$[\pi/4, 0]$
$n\|_{\text{optimum}}$	0	1	2
$\Phi_\alpha\|_{\text{optimum}}$	$\pi - \theta_m$	$\pi/2 - \theta_m$	$-\theta_m$
Medial axis appearance			
Cross-field behaviour			
Colour			

Figure 5: Curve between two boundary-aligned crosses of nearby boundaries (left) and a table summarising the optimum mesh-flow types for medial angle ranges (right).

cross-field is the most reasonable default target in the absence of supplementary information on the local target size and orientation of the mesh. This equates to looking for a mesh with minimal element distortion. With this target the total flux should be minimised and the optimum integer for n can be computed by

$$n|_{\text{optimum}} = \text{round}\left(\frac{\int_\alpha \kappa_g \, ds + \theta_b - \theta_a}{\pi/2}\right). \tag{5}$$

For the particular case of a planar surface with the curve α composed of the medial radii (traversed so that κ_g is positive), then $\int_\alpha \kappa_g \, ds = \pi - \theta_m$. If the touching points are on boundary edges then θ_a and θ_b are zero, therefore

$$n|_{\text{optimum}} = \text{round}\left(\frac{\pi - \theta_m}{\pi/2}\right). \tag{6}$$

The three basic mesh-flow types between boundary edges linked by the medial axis and their preferred medial angle ranges are summarised in the table in Fig. 5 (right). It may be helpful to think of the medial axis being aligned with the cross-field for $n_c = 0$ and $n_c = 2$ and running diagonally through the cross-field for $n_c = 1$.

If the touching points are on concave vertices θ_a and θ_b are not zero in general. To determine the optimum mesh-flow type in these cases the cross directions from which θ_a and θ_b are measured are chosen so that their values are minimised, and the non-zero entry $\theta_b - \theta_a$ is included in the calculation of Φ_α. Alternatively, the medial radii directions can be adjusted by θ_a and θ_b before θ_m is measured as shown in Fig. 6 (left).

The optimum mesh patterns at boundary corners can be decided by minimising the associated total flux radiating from the corner in the ϕ-field. Bunin gives an analytic expression for the strength of a point source necessary at a corner which involves the corner angle, θ_c, and an integer, n_c, that specifies how many elements occur there. Choosing n_c to minimise the strength gives the optimum ranges shown in Fig. 6 (right).

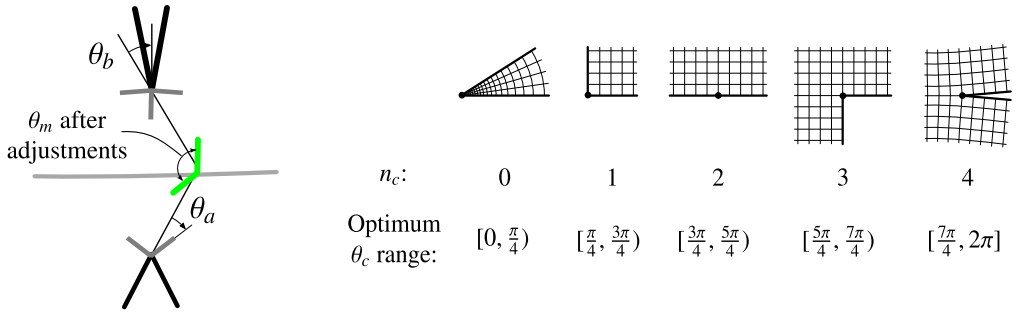


Figure 6: Adjustments at concave vertices (left) and optimum mesh patterns at corners (right).

The cross-field degenerates at a corner vertex in the same way as it does at a mesh singularity. At a corner vertex the cross-field has a number of directions which are always evenly distributed angularly. Except for $n_c = 0$ (which is especially degenerate), the cross-field has two directions pointing along the edges forming the corner and $(n_c - 1)$ directions pointing into the surface.

3.2. Identifying mesh singularities

If the optimum mesh-flow behaviour is selected for every position of the medial axis the positions of mesh singularities are in effect decided. A general method for identifying these positions involves a flux balance calculation. Supposing the surface is sliced into a series of thin regions by medial radii as shown in Fig. 7, 8 and 9. Treating each sliver as a control area, a flux balance analysis can be carried out using the medial angle-dependent values for the

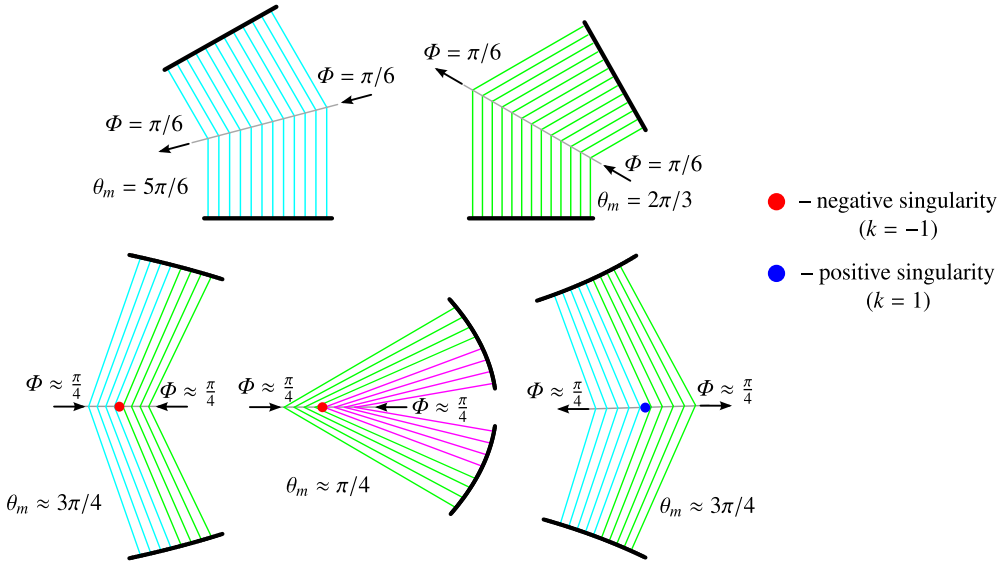


Figure 7: Examples of flux balancing along medial edges between smooth boundaries.

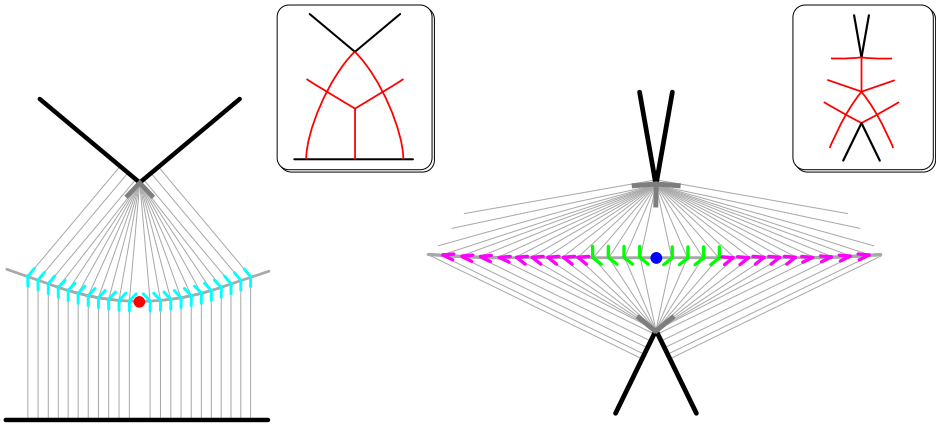


Figure 8: Examples of flux balancing along medial edges between boundaries with concave corners. Sketches of the implied mesh topologies are shown in the boxes.

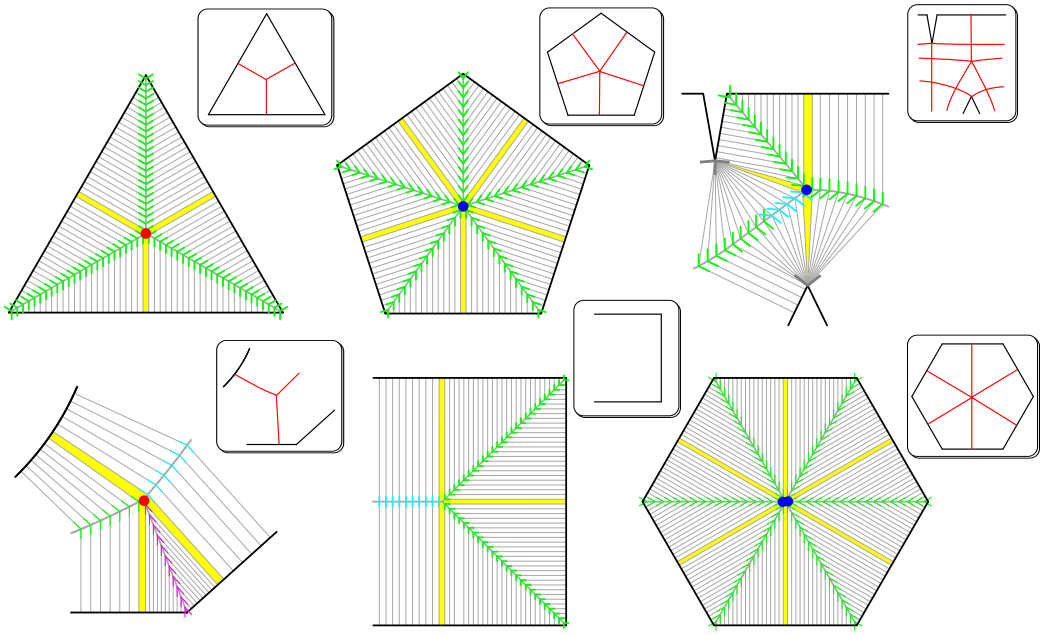


Figure 9: Examples of flux balancing at medial vertices (highlighted).

total fluxes across pairs of medial radii as described above. The total fluxes across the small boundary segments are given by the differences in angle of the adjacent boundary-aligned crosses (which should be small). The flux residual must equal $k\pi/2$ where the integer k denotes the type of singularity present. Most usually only simple singularities occur, *i.e.* $k = \pm 1$ or 'positive' and 'negative' singularities. In some special cases higher-order singularities can occur, an example of which is at the central medial vertex of a regular hexagon, as shown in Fig. 9 (bottom right) where a $k = 2$ type singularity occurs. This is identified in a general fashion.

At a medial vertex where the touching circle has a finite contact with a boundary there are a range of medial angle values associated with the medial vertex. These features can be equated to the limiting case where the boundary curve tends to a circular arc from an elliptical arc. Thus, the singularity type at the medial vertex is given by

$$k = -\text{floor}\left(\frac{\max(\theta_m)}{\pi/2}\right). \tag{7}$$

For example, a semi-circular end of a rectangular block has a corresponding

medial vertex with $\max(\theta_m) = \pi$. Therefore, a $k = -2$ singularity is placed on the medial vertex. This is illustrated in Fig. 10.

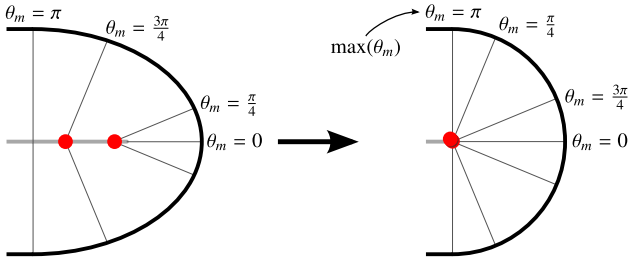


Figure 10: Treatment of finite contact regions. A circular arc is equivalenced to an elliptical arc whose curvature tends towards uniformity.

3.3. Algorithm

A basic algorithm was developed to use the principles just described to find an effective arrangement of mesh singularities. Its general steps are outlined in Alg. 1 below. It was implemented in a Python program using the CADfix API [31].

Algorithm 1. Find mesh singularities using medial axis

1. Generate the medial axis for the surface. CADfix's [31] medial axis generation tool is used for this.
2. For each medial edge, assemble a sequence of analysis positions between the start and end points of the edge. A suitable interval size will capture the smooth change in θ_m and ensure that a single critical switch point of the optimum mesh-flow is contained in the interval.
3. Perform a total flux balance calculation for each sliver region bounded between the interval positions on medial edges and medial vertices, to determine appropriate singularity positions.

The essential output of the algorithm is a layer of information on the medial axis detailing the mesh-flow type between associated boundaries and the positions where singularities occur. Line geometry is also created purely to illustrate the method which is shown in the next section.

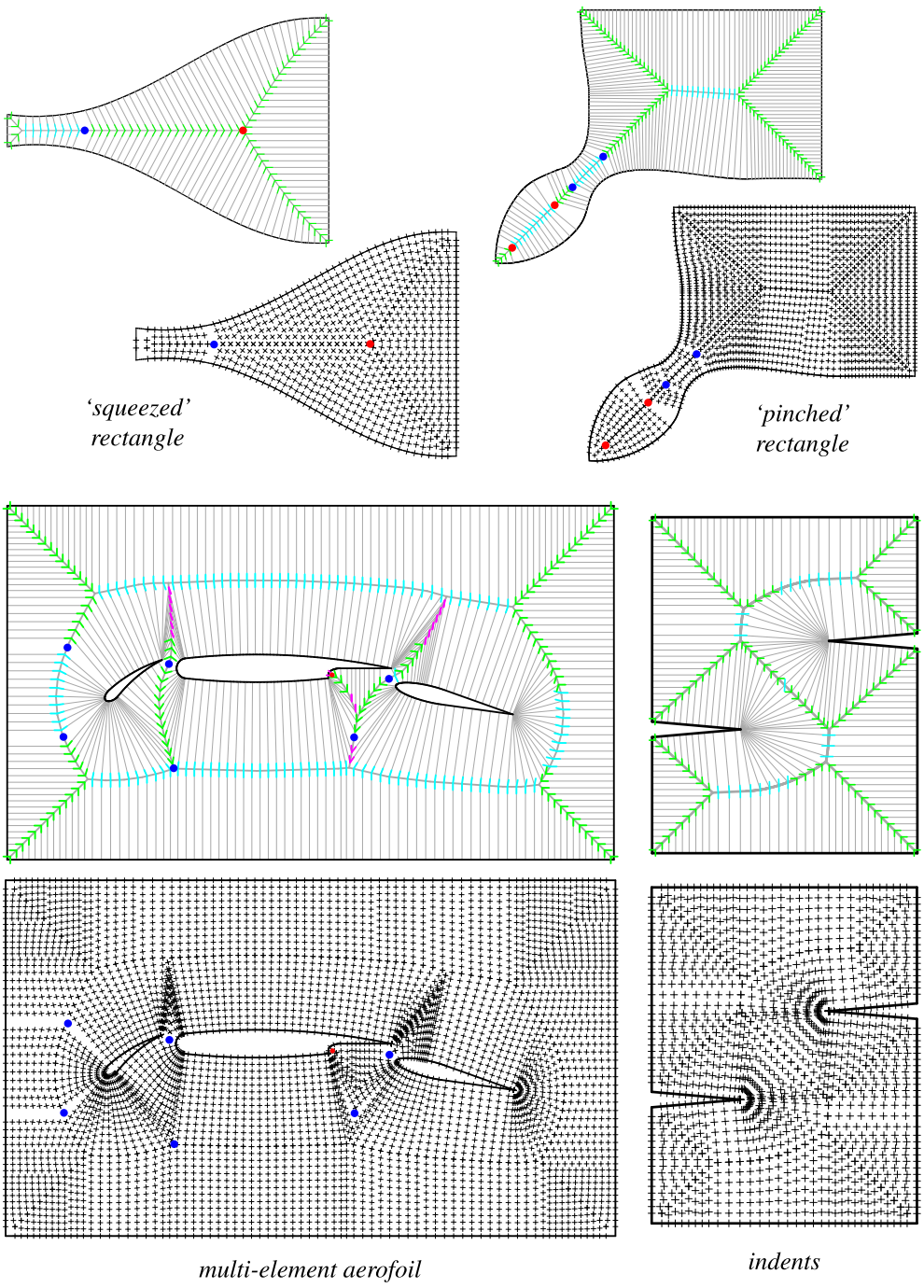


Figure 11: Example results of singularity identification algorithm.

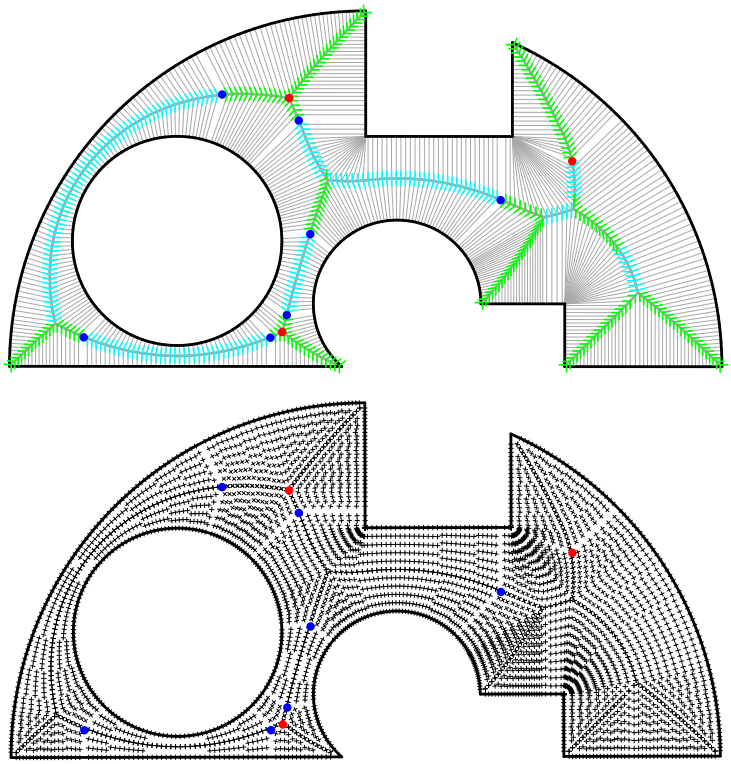


Figure 12: Result of singularity identification algorithm for *test surface 1*.

3.4. Singularity identification results

Some illustrative results of the algorithm are shown Figs. 11 and 12. Two images are included for each example, one showing the medial axis and artefacts representing the solution process, and the other showing a crudely defined cross-field. When creating the crosses their overall turning behaviour over the medial radii is taken from the assigned mesh-flow type and a smooth and even rate of change of angle along the medial radii is assumed.

The geometries in Fig. 11 were seen previously in Figs. 2–4. The results match the 'preferred decomposition' topologies in each case. The additional positive and negative singularities appear in their natural positions in the *'squeezed'* and *'pinched' rectangle* surfaces. And, the concavities in the *multi-element aerofoil* and *indents* surfaces are managed nicely. The result in Fig. 12 for *test surface 1* contains some collections of singularities in close proximity to each other. Depending on the target coarseness of the quad mesh these should be merged together to cancel each other out (for positive

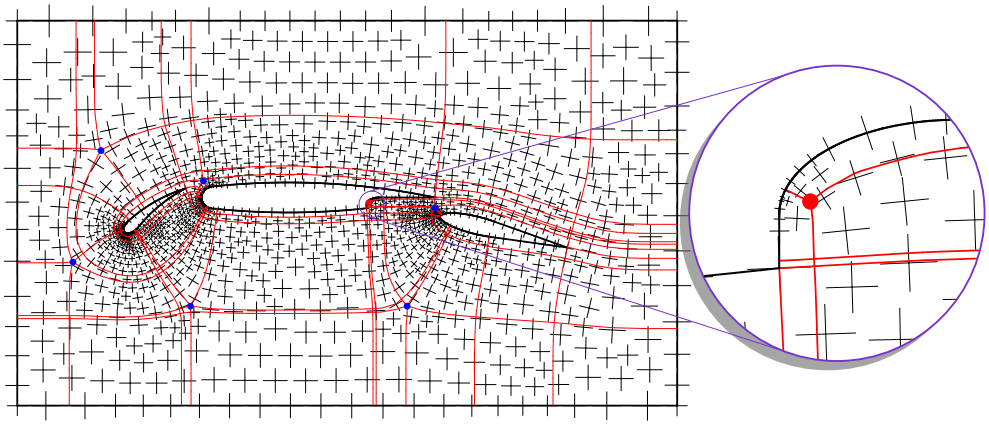


Figure 13: Multiblock decomposition of *multi-element aerofoil* by [32].

and negative singularity pairs) or to create higher-order singularities (for singularities of the same sign). The proposed decomposition method described in the next section achieves this.

One approach to generating a multiblock decomposition is to trace the separatrices of a cross-field. This has been explored in [19, 32, 20] and robust and effective cross-field initialisation and streamline tracing on a triangulation of the surface have been developed. A decomposition of the *multi-element aerofoil* surface is shown in Fig. 13 that was created by the method of [32] with a cross-field and singularities that correspond with those in Fig. 11. In principle, these methods could be applied in the present method (with some modifications to clean up the cross-fields) but it would eliminate the niche that the medial axis decomposition methods fit into. Medial-axis-based methods should be 'lightweight', cheap, fast and robust, otherwise they offer no advantage over more sophisticated methods [20, 21, 22] that can handle more general problems with target size and orientation fields. To fulfil these objectives undemanding techniques are called for to split the geometry using the medial axis in a similar manner to the T&A and TopMaker methods, but that can produce enhanced decompositions. The splits should follow the implicit mesh solution already achieved.

3.5. Rethinking the algorithm design

The algorithm as laid out in Alg. 1 is simplistic and easy to understand. Moreover, it is easy to program. However, a more streamlined algorithm is

possible which does not use a fixed set of analysis positions on the medial axis to define small control areas on which to compute flux balances. After assigning optimum mesh flow across the medial axis, flux imbalances, hence singularity occurrences, can only happen at three types of position on the medial axis: (1) points with critical θ_m values of $\pi/4$ and $3\pi/4$ with touching points on smooth boundary edges, (2) medial vertices and (3) points along sections of the medial axis with a touching point on a concave corner where the cross-field direction for measuring the adjusted θ_m switches from one to another. This means that only these points really need to be assessed. So, a more efficient algorithm would concentrate on identifying these important analysis points on the medial axis and simply check them to find mesh singularities.

4. Decomposition

The primary objective of the decomposition process is to create *logically* convex polygon subregions around the mesh singularities and for the remaining subregions to be submappable. The adjective *logical* is used to indicate a property that is distinguished after the mesh is locally mapped to a Cartesian form with all edges along the coordinate directions. Logically convex m-sided polygons can be meshed using the simple templates shown in Fig. 14. The 3+-sided polygon templates are the applications of mid-point subdivision. The 0-sided template is a widely used topology for meshing circles (see e.g. [33]) and 1- and 2-sided templates are proposed simple candidates. Submappable regions can be meshed by a number of

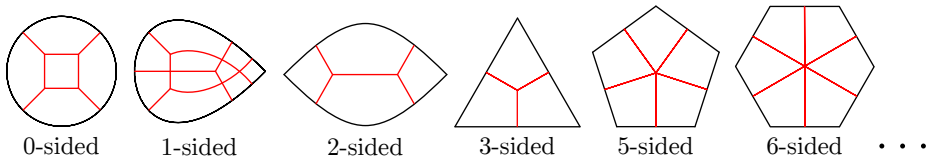


Figure 14: Mesh templates for logically convex m-sided polygons.

methods [13, 34, 35]. Two examples of submapped meshes are shown in Fig. 15.

By not fixing the singularity positions they are free to be repositioned inside their polygon subregions to suit the target element sizes and therefore allow better quality meshes to be produced. However, the proposed method

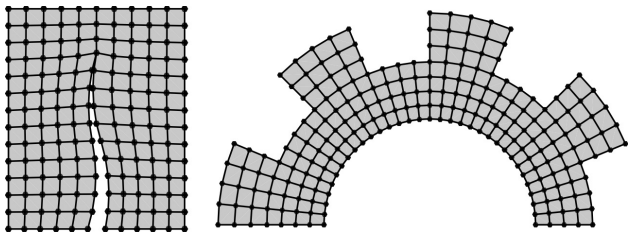


Figure 15: Submapped meshes (taken from [34]).

cannot always succeed in creating a polygon subregion around a mesh singularity. Sometimes, its position is fixed instead and it appears as a corner vertex of adjacent subregions.

4.1. Decomposition splits

The medial axis links proximate boundaries and it makes available candidate splits between the boundaries. For two boundary edges coupled by the medial axis, the medial radii spanning between them can be used as decomposition splits. There are two standard types that are serviceable: $n = 0$ and $n = 1$ medial radii pairs as shown in Fig 16. The $n = 0$ type split is

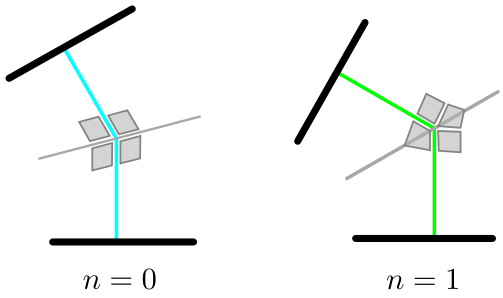


Figure 16: Decomposition splits between boundary edges.

preferable as it does not create a concave corner and medial angles closer to π and $\pi/2$ are more favourable for each respectively.

For a logically concave vertex (*i.e.* $n_c = 3$ or 4) there are a discrete number of medial radii connected to the vertex that are allowed to be used as part of a split. These radii are either exactly aligned with the cross-field directions at the corner vertex, or are almost aligned and perpendicular to the edges forming the corner. This is illustrated in Fig. 17.

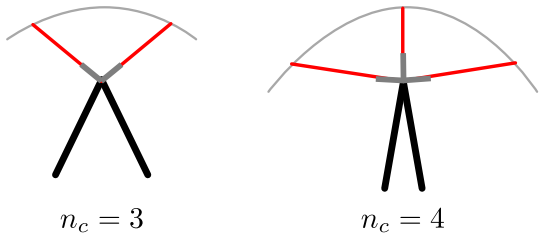


Figure 17: Decomposition edges from concave corners.

The most complex scenario is when two concave vertices are on opposite sides of the medial axis. In these cases, there are no legitimate splits composed of pairs of medial radii generally. A simple work-around is to join a splitting edge of one orthogonally to the most suitable opposite splitting edge. This is illustrated in Fig. 18. These splits can be determined as follows:

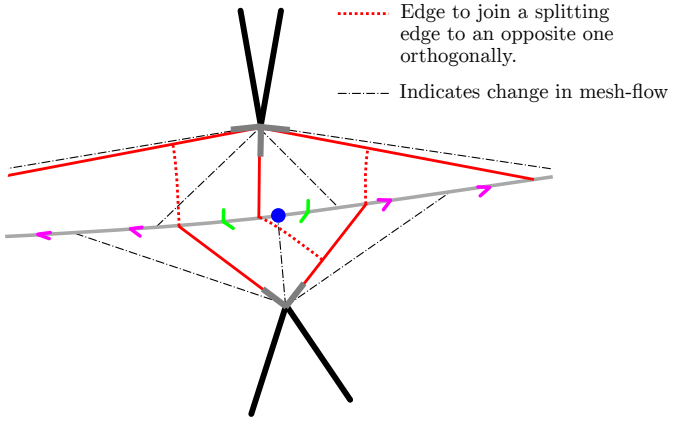


Figure 18: Illustration of the construction of potential decomposition splits between two concave corners.

- At a concave vertex, for one of its cross-field directions find the corresponding point on the medial axis.
- If the opposite touching point is on an edge then simply use the medial radii as the second half of the split.
- If the opposite touching point is also a concave vertex:
  - If $n = 0$ or $n = 1$ find a projection to one of the opposite cross-field directions *along* the adjusted medial radius direction.

– If $n = 2$ find a projection to one of the opposite cross-field directions *perpendicular to* the adjusted medial radius direction.

4.2. Decomposition process details

A running example will be used to demonstrate the decomposition process steps on *test surface surface 1* as they are described.

Step 1. Potential splits are determined for concave corners. The splits are named 'CCi_j' where i is a number identifier for the concave corner and j is the split number connected to it. These can be seen in Fig. 19 as labelled dotted lines.

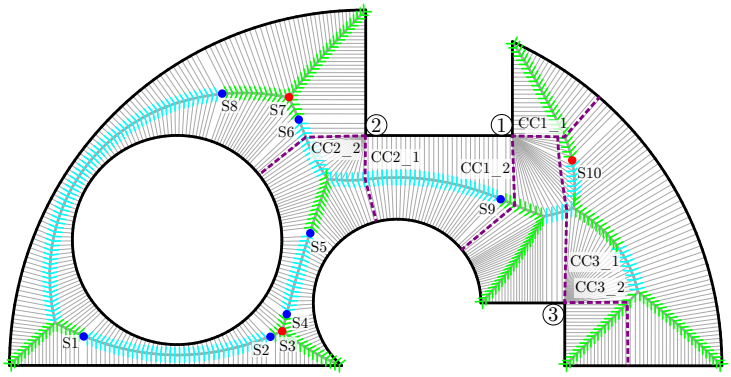


Figure 19: Potential decomposition splits from concave corners for *test surface 1.*

Step 2. A database is assembled for storing a list of *neighbouring features* for each singularity. All singularities are named ('Si') and each is assessed in turn (in a non-specific order) as the *centre singularity* with its name as the *key* to the associative array or hash table. The medial axis is explored in all directions from the centre singularity position. Once another singularity, convex corner or concave corner split is encountered the exploration of that medial axis branch is stopped and a descriptive identifier is added to the neighbouring feature list:

- For a neighbouring singularity its name is added.
- For a convex corner an identifier – CX – followed by its n_c type is added.
- For a concave corner split its name is added.

singularity	sign	neighbouring features
S1	+1	(S2,0), (S8,0), (*CX1*, 1)
S2	+1	(S1,0), (S3,0)
S3	−1	(S2,0), (S4,0), (*CX1*, 1)
S4	+1	(S3,0), (S5,0)
S5	+1	(S4,0), (CC2_1,1), (CC2_2,1)
S6	+1	(CC2_2, 1), (S7, 0)
S7	−1	(S6,0), (S8, 0), (*CX1*, 1)
S8	+1	(S7, 0), (S1,0), (*CX1*, 1)
S9	+1	(CC2_1,1), (CC1_2,1)
S10	−1	(CC1_1, 0), (CC3_1,1), (CC3_2, 0), (*CX1*, 1)

Table 1: The initialised database for *test surface 1*.

An additional Boolean attribute is added with these to mark whether the neighbouring feature is 'unresolved' (0) or 'resolved' (1), indicating whether or not a split needs to be created between it and the centre singularity in order to produce a convex polygon. The possibilities are as follows:

- If a neighbouring concave corner split creates a logically straight edge or convex corner with respect to the centre singularity it is considered resolved.
- If a neighbouring convex corner is of type $n_c = 0$ (*CX0*) it is marked unresolved, otherwise it is marked resolved.
- A neighbouring finite contact medial axis vertex without a singularity is marked as resolved.
- A neighbouring singularity is marked as unresolved.

The initialised database for *test surface 1* is shown in Tab. 1.

Step 3. Each singularity is dealt with in turn as the centre singularity of the evaluation, with the objective of creating decomposition splits around it to produce a logically convex polygon. If all the neighbouring features are resolved then this has already been achieved. For each unresolved neighbouring feature a split is searched for which would create a logically straight corner ($n = 0$) across the intermediate portion of the medial axis, between it and the centre singularity. If the unresolved neighbouring feature is not a singularity then splits that create logically convex corners (from the perspective of the centre singularity) are also permitted. The candidate splits

that are found are ranked in order to choose the best one where $n = 0$ types are preferred over $n = 1$ and medial angles closer to π and $\pi/2$ are more favourable for each. If no permitted split can be found between the centre singularity and a neighbouring feature then the centre singularity needs to be *fixed*, or if the neighbouring feature is a singularity then it may by fixed instead.

Fixing a singularity involves putting a split through it with the result that it appears as a corner vertex of subregion blocks in the decomposition, as shown in Fig. 20. It is possible to create the complete decomposition by

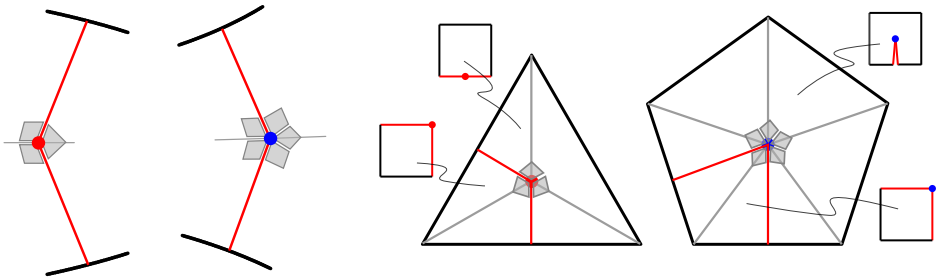


Figure 20: Examples of fixing singularities by creating splits through them. (Logical shapes of subregions are also shown in the two examples on the right)

fixing every singularity, thus generating submappable subregions. However, this strategy would create a complex decomposition with a large number of subregions. Also, it is more desirable to allow singularities to float around inside m-sided polygons to suit even distributions of element sizes. For these reasons singularities are only fixed when necessary.

If the central singularity has only unresolved neighbouring features which are singularities, it is checked to see if all the neighbouring singularities can be fixed to create a logically convex subregion around the centre singularities. If so, then this is carried out. Otherwise the centre singularity is fixed. In every case the database is updated to record whether the adjacent singularities can be considered resolved.

4.2.1. Decomposition process example

The route by which the decomposition process is performed for *test surface 1* is illustrated in Fig. 21. The order in which singularities are dealt with as the centre singularity follows the arbitrarily assigned numbering. There may be slight variation in the decomposition that is created depending on the arbitrary numbering. However, almost identical meshes should be able

to be generated on the possible decompositions. The changes in the database at each point in the process are shown in Fig. 22. The process is carried out as follows:

1. With S1 as the centre singularity the two unresolved neighbouring features are S2 and S8. Decomposition splits of type $n = 0$ are found that can resolve these. Since $n = 0$ type decomposition splits create logically flat edges, S1 can be considered resolved as a neighbouring feature for S2 and S8.

2. With S2 as the centre singularity the only unresolved neighbouring feature is S3. However, there is no permitted split between S2 and S3. The first check is to see whether S3 can be fixed to create a logically flat or convex corner from the perspective of S2. It is found that S3 can be fixed to create a logically convex corner for S2. Since S3 is at a medial vertex it is prudent to see if additional decomposition edges can be created so that it can be made resolved for its other neighbouring features. Thus, the third edge is created to produce a logically convex corner from the perspective of S4. S3 has been fixed so nothing remains to be done for it.

3. With S4 as the centre singularity a logically flat (n=0) split is found between it and S5.

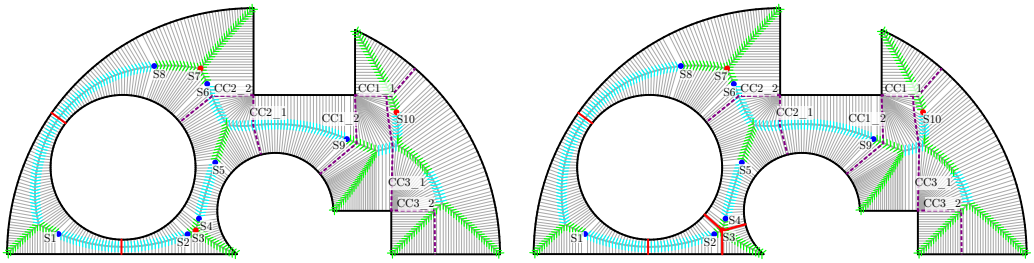
4. With S5 as the centre singularity all neighbouring features are resolved after setting the concave corner splits CC2_1 and CC2_2.

5. With S6 as the centre singularity the only unresolved singularity is S7. S7 is fixed and is made resolved as a neighbouring feature of both S6 and S8. S8 now has only resolved neighbouring features so nothing remains to be done for it.

6. With S9 as the centre singularity all neighbouring features are resolved after setting the concave corner split CC1_2.

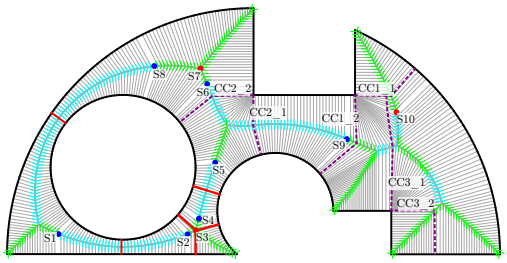
7. With S10 as the centre singularity a split cannot be found to resolve CC_1. Therefore S10 is fixed.

The final decomposition and a mesh that is generated on it by a combination of midpoint subdivision and submapping methods is shown in Fig. 21. Mesh singularities are perceivable in the subregions following the implicit medial axis mesh-flow solution but not necessarily in the exact same positions.

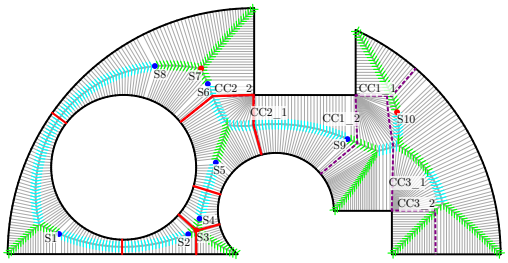


1

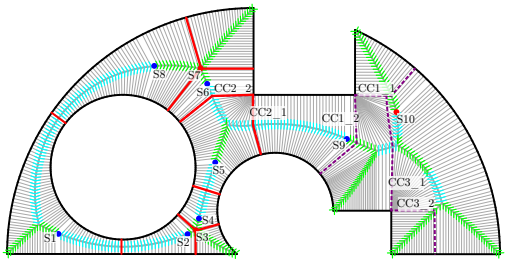
2



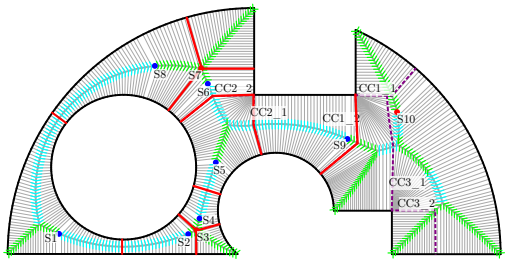
3



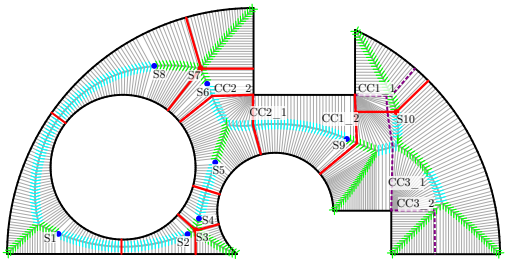
4



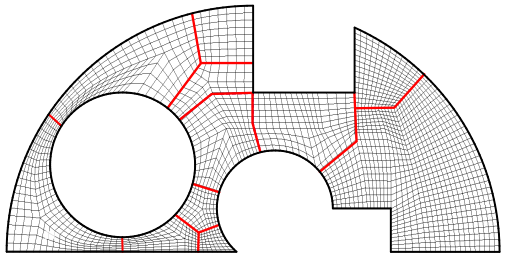
5



6



7



Final decomposition and mesh

Figure 21: Demonstration of decomposition process on *test surface 1.*

singularity	sign	neighbouring features
S1	+1	(S2,1), (S8,1), (*CX1*, 1)
S2	+1	(S1,1), (S3,0)
S3	−1	(S2,0), (S4,0), (*CX1*, 1)
S4	+1	(S3,0), (S5,0)
S5	+1	(S4,0), (CC2_1,1), (CC2_2,1)
S6	+1	(CC2_2, 1), (S7, 0)
S7	−1	(S6,0), (S8, 0), (*CX1*, 1)
S8	+1	(S7, 0), (S1,1), (*CX1*, 1)
S9	+1	(CC2_1,1), (CC1_2,1)
S10	−1	(CC1_1, 0), (CC3_1,1), (CC3_2, 0), (*CX1*, 1)

1

singularity	sign	neighbouring features
S1	+1	(S2,1), (S8,1), (*CX1*, 1)
S2	+1	(S1,1), (S3,1)
S3	−1	(S2,1), (S4,1), (*CX1*, 1)
S4	+1	(S3,1), (S5,0)
S5	+1	(S4,0), (CC2_1,1), (CC2_2,1)
S6	+1	(CC2_2, 1), (S7, 0)
S7	−1	(S6,0), (S8, 0), (*CX1*, 1)
S8	+1	(S7, 0), (S1,1), (*CX1*, 1)
S9	+1	(CC2_1,1), (CC1_2,1)
S10	−1	(CC1_1, 0), (CC3_1,1), (CC3_2, 0), (*CX1*, 1)

2

singularity	sign	neighbouring features
S1	+1	(S2,1), (S8,1), (*CX1*, 1)
S2	+1	(S1,1), (S3,1)
S3	−1	(S2,1), (S4,1), (*CX1*, 1)
S4	+1	(S3,1), (S5,1)
S5	+1	(S4,1), (CC2_1,1), (CC2_2,1)
S6	+1	(CC2_2, 1), (S7, 0)
S7	−1	(S6,0), (S8, 0), (*CX1*, 1)
S8	+1	(S7, 0), (S1,1), (*CX1*, 1)
S9	+1	(CC2_1,1), (CC1_2,1)
S10	−1	(CC1_1, 0), (CC3_1,1), (CC3_2, 0), (*CX1*, 1)

3

singularity	sign	neighbouring features
S1	+1	(S2,1), (S8,1), (*CX1*, 1)
S2	+1	(S1,1), (S3,1)
S3	−1	(S2,1), (S4,1), (*CX1*, 1)
S4	+1	(S3,1), (S5,1)
S5	+1	(S4,1), (CC2_1,1), (CC2_2,1)
S6	+1	(CC2_2, 1), (S7, 0)
S7	−1	(S6,0), (S8, 0), (*CX1*, 1)
S8	+1	(S7, 0), (S1,1), (*CX1*, 1)
S9	+1	(CC2_1,1), (CC1_2,1)
S10	−1	(CC1_1, 0), (CC3_1,1), (CC3_2, 0), (*CX1*, 1)

4

singularity	sign	neighbouring features
S1	+1	(S2,1), (S8,1), (*CX1*, 1)
S2	+1	(S1,1), (S3,1)
S3	−1	(S2,1), (S4,1), (*CX1*, 1)
S4	+1	(S3,1), (S5,1)
S5	+1	(S4,1), (CC2_1,1), (CC2_2,1)
S6	+1	(CC2_2, 1), (S7, 1)
S7	−1	(S6,1), (S8, 1), (*CX1*, 1)
S8	+1	(S7, 1), (S1,1), (*CX1*, 1)
S9	+1	(CC2_1,1), (CC1_2,1)
S10	−1	(CC1_1, 0), (CC3_1,1), (CC3_2, 0), (*CX1*, 1)

5

singularity	sign	neighbouring features
S1	+1	(S2,1), (S8,1), (*CX1*, 1)
S2	+1	(S1,1), (S3,1)
S3	−1	(S2,1), (S4,1), (*CX1*, 1)
S4	+1	(S3,1), (S5,1)
S5	+1	(S4,1), (CC2_1,1), (CC2_2,1)
S6	+1	(CC2_2, 1), (S7, 1)
S7	−1	(S6,1), (S8, 1), (*CX1*, 1)
S8	+1	(S7,1), (S1,1), (*CX1*, 1)
S9	+1	(CC2_1,1), (CC1_2,1)
S10	−1	(CC1_1, 0), (CC3_1,1), (CC3_2, 0), (*CX1*, 1)

6

singularity	sign	neighbouring features
S1	+1	(S2,1), (S8,1), (*CX1*, 1)
S2	+1	(S1,1), (S3,1)
S3	−1	(S2,1), (S4,1), (*CX1*, 1)
S4	+1	(S3,1), (S5,1)
S5	+1	(S4,1), (CC2_1,1), (CC2_2,1)
S6	+1	(CC2_2, 1), (S7, 0)
S7	−1	(S6,0), (S8, 0), (*CX1*, 1)
S8	+1	(S7, 0), (S1,1), (*CX1*, 1)
S9	+1	(CC2_1,1), (CC1_2,1)
S10	−1	(CC1_1, 0), (CC3_1,1), (CC3_2, 0), (*CX1*, 1)

7

Figure 22: Database changes during the decomposition process on *test surface 1.*

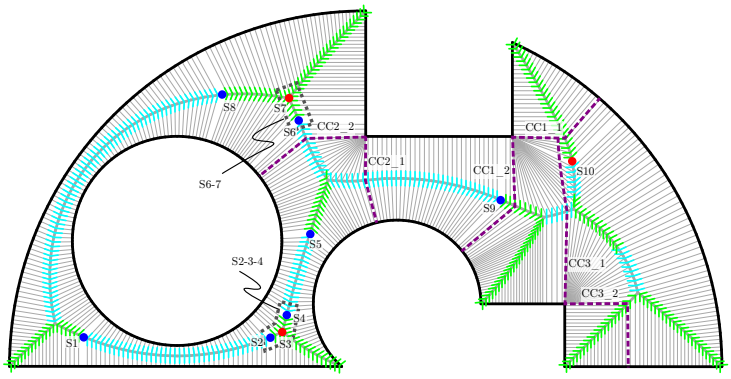


Figure 23: Merging of singularities for *test surface 1*

singularity	sign	neighbouring features
S1	+1	(S2-3-4,0), (S8,0), (*CX1*, 1)
S2-3-4	+1	(S1,0), (*CX1*, 1), (S5,0)
S5	+1	(S2-3-4,0), (CC2_1,1), (CC2_2,1)
S6-7	0	(CC2_2, 1), (S8,0), (*CX1*, 1)
S8	+1	(S6-7, 0), (S1,0), (*CX1*, 1)
S9	+1	(CC2_1,1), (CC1_2,1)
S10	−1	(CC1_1, 0), (CC3_1,1), (CC3_2, 0), (*CX1*, 1)

Table 2: The initialised database for *test surface 1* after singularities are merged.

4.3. Merging singularities

In the previous example the decomposition has been carried out with the intention that all singularities that are not exactly superimposed should be separated. However, it is usually preferable if they are merged together when they are below a minimum distance tolerance of each other. Ideally, the local target mesh size should dictate the tolerance. A geometry-based alternative is to use a fraction of the local medial radius to set the local tolerance value. A reasonable range for the tolerance is $[r_m/4, r_m]$.

4.4. Decomposition example with merging of singularities

The singularities which are to be grouped together and merged for the tolerance setting of $3r_m/4$ are illustrated in Fig. 23. The database is modified by treating the grouped singularities as a single combined singularity and updating the affected entries to register that, as shown in Tab. 2.

Since S6 has a sign of +1 and S7 has a sign of −1, the combined singularity, S6-7, has a net sign of 0. Therefore, it is a regular grid point and it

singularity	sign	neighbouring features
S1	+1	(S2-3-4,0), (S8,0), (*CX1*, 1)
S2-3-4	+1	(S1,0), (*CX1*, 1), (S5,0)
S5	+1	(S2-3-4,0), (CC2_1,1), (CC2_2,1)
S8	+1	(S6-7, 0), (S1,0), (CC2_1,1), (*CX1*, 1), (*CX1*, 1)
S9	+1	(CC2_1,1), (CC1_2,1)
S10	−1	(CC1_1, 0), (CC3_1,1), (CC3_2, 0), (*CX1*, 1)

Table 3: The database for *test surface 1* after singularities are merged and S6-7 is removed.

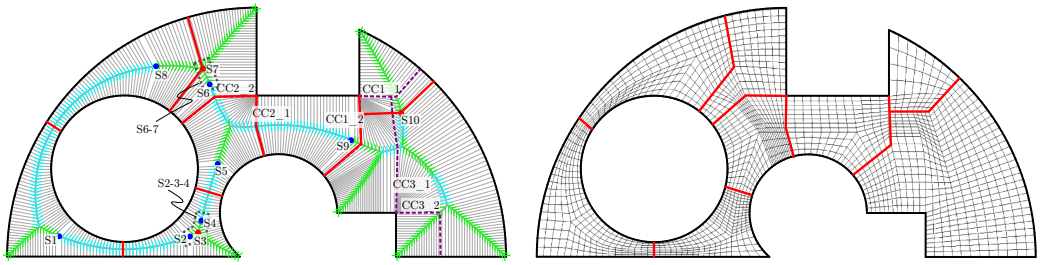


Figure 24: Final decomposition and mesh after treating proximate singularities as merged.

doesn't need to be treated as a singularity in the decomposition process. It is removed from the database and the entries for adjacent singularities are updated, as shown in Tab. 3.

There are now fewer singularities to deal with and consequently the decomposition process requires fewer steps and the decomposition that is produced following the same procedure described previously is simpler. The final decomposition with an example mesh is shown in Fig. 24.

4.5. Decomposition examples

Some examples showing the decomposition and example meshes thereon for a selection of surfaces are shown in Fig. 25. The quality of the meshes could be boosted by applying smoothing methods, but this has not been done in order to preserve the decomposition edges in the mesh for clarity.

The 'preferred decompositions' which were introduced in Sec. 1.3.1 are achieved for the *'squeezed' rectangle* and *'pinched' rectangle* surfaces.

A decomposition is not shown for the *indents* surface (Fig. 3) because the solution given by the method does not contain any singularities (Fig. 11), which corresponds with its 'preferred decomposition'. Therefore, submapping can be applied directly to generate a mesh.

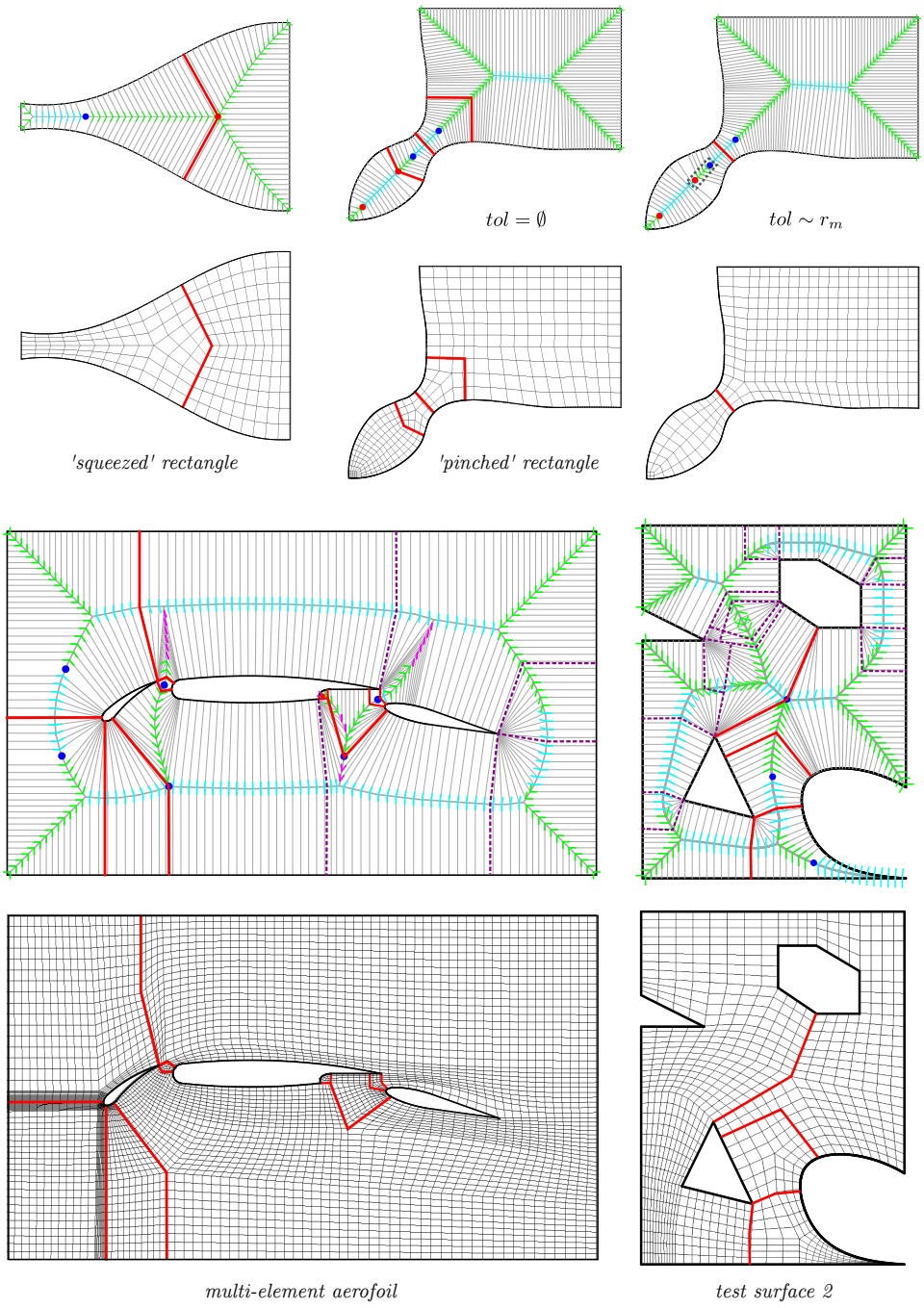


Figure 25: Decomposition examples

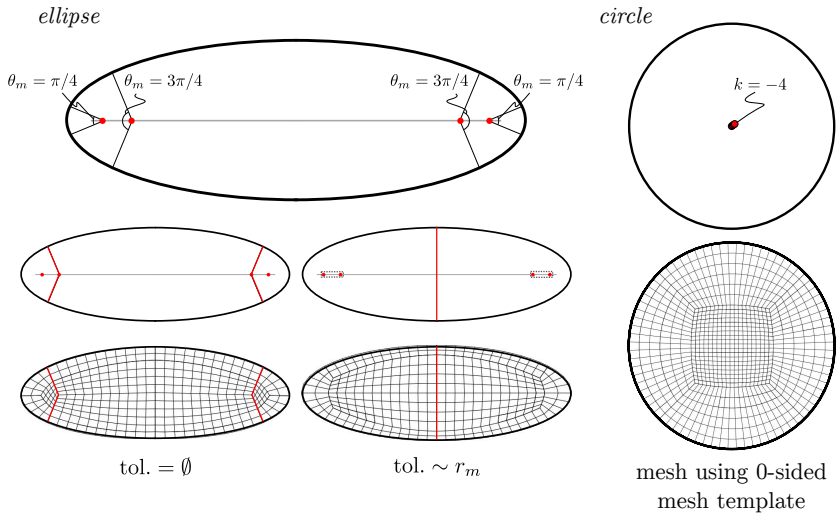


Figure 26: *Ellipse* and *circle* decomposition examples

A decomposition is created for the *multi-element aerofoil* which facilitates a more smoothly varying mesh than that created by the TopMaker and TA algorithms (seen previously in Fig. 2).

The *test surface 2* case is a relatively simple geometry but it contains a number of holes and concavities which can cause issues with other decomposition methods. The proposed method produces a pleasing decomposition. The degenerate mesh at the lower right hand corner of the surface may draw disapproval but, theoretically speaking, it is the best way of meshing such an osculating corner.

Other decomposition examples for an *ellipse* and a *circle* surface are shown in Fig. 26. Singularities are found at critical angles of $\pi/4$ and $3\pi/4$ with a short distance between them on the ellipse. Using a low tolerance setting, a decomposition is generated with a topology according to the inferred medial axis solution (left). But with a higher tolerance the ellipse is divided into two 2-sided subregions (right).

The medial axis for a circle is a single point at its centre with a touching circle in complete finite contact. This means that there are four superimposed negative singularities in the solution. No decomposition is necessary and the circle is meshed using the 0-sided template.

4.6. Additional final splitting of multiply connected subregions

In some cases where the surface contains holes additional splits may be required in order to produce submappable regions which are topological discs with a single closed loop of boundary edges. An example would be an annulus which the proposed method would leave as is since a 'radial grid' of concentric mesh rows is a good solution. However, if only mapping and submapping algorithms are available then it should be decomposed into a rectangle by creating a decomposition split between the inner and outer boundary loops. It is not difficult to add an extra step to the end of the procedure to check if such splits are required to convert multiply connected subregions to simply connected subregions and to find appropriate splits across the medial axis.

5. Conclusion

A novel approach has been described for using the medial axis to first identify an effective configuration of mesh singularities, and secondly, to decompose the surface for block-structured meshing. The weaknesses of existing medial axis based methods due to assumed mesh behaviour in relation to the medial axis which is inappropriate around concavities are avoided. The proposed methods offer more capable handling of 2-D surfaces of all types for generating high-quality decompositions. And yet, the increased generality does not entail a large increase in complexity.

Future work involves refining the implementation details of the singularity identification and decomposition algorithms. This is not trivial work and substantial effort will be required to develop a reliable and efficient method. However, because the presented methods are based on a solid theory and the rationale is clear for every step of the described algorithms, there should be no major problems lurking in the practical details of a general implementation.

Acknowledgements

The authors acknowledge the support of ARA and the Technology Strategy Board (TSB) for funding within the Airbus Numerical Simulation and Design (ANSD) Project. The authors would also like to thank TranscenData for their support.

References

[1] Z. Ali, P. G. Tucker, Multiblock structured mesh generation for turbomachinery flows, in: J. Sarrate, M. Staten (Eds.), Proceedings of the 22nd International Meshing Roundtable, Springer International Publishing, 2013, pp. 165–182. doi:10.1007/978-3-319-02335-9_10.

[2] L. Nackman, V. Srinivasan, Method of generating finite elements using the symmetric axis transform, US Patent 4,797,842 (1989).
URL `http://www.google.co.uk/patents/US4797842`

[3] T. K. H. Tam, C. G. Armstrong, 2-D finite element mesh generation by medial axis subdivision, Advances in Engineering Software 13 (5/6) (1991) 313–324.

[4] E. Catmull, J. Clark, Recursively generated b-spline surfaces on arbitrary topological meshes, Computer-Aided Design 10 (6) (1978) 350 – 355. doi:10.1016/0010-4485(78)90110-0.

[5] T. K. H. Tam, C. G. Armstrong, Finite element mesh control by integer programming, International Journal for Numerical Methods in Engineering 36 (15) (1993) 2581–2605. doi:10.1002/nme.1620361506.

[6] T. Li, R. McKeag, C. Armstrong, Hexahedral meshing using midpoint subdivision and integer programming, Computer Methods in Applied Mechanics and Engineering 124 (12) (1995) 171 – 193. doi:10.1016/0045-7825(94)00758-F.

[7] Abaqus FEA, SIMULIA website. Dassault Systemes., `http://www.3ds.com/products-services/simulia/portfolio/abaqus/`, (Accessed: 12/6/2013).

[8] D. Rigby, TopMaker: A technique for automatic multi-block topology generation using the medial axis, Tech. Rep. NASA/CR–213044 (2004).

[9] D. Guoy, J. Erickson, Automatic Blocking Scheme for Structured Meshing in 2D Multiphase Flow Simulation, in: Proceedings, 13th International Meshing Roundtable, 2004-3765C, Williamsburg, VA, 2004, pp. 121–132.

[10] H. Xia, P. G. Tucker, Fast equal and biased distance fields for medial axis transform with meshing in mind, Applied Mathematical Modelling 35 (12) (2011) 5804 – 5819. doi:10.1016/j.apm.2011.05.001.

[11] J. F. Thompson, B. K. Soni, N. P. Weatherill (Eds.), Handbook of Grid Generation, Part 1. Block-Structured Grids, 1st Edition, CRC Press, 1998.

[12] T. Li, C. Armstrong, R. McKeag, Quad mesh generation for k-sided faces and hex mesh generation for trivalent polyhedra, Finite Elements in Analysis and Design 26 (4) (1997) 279 – 301. doi:10.1016/S0168-874X(96)00085-6.

[13] D. White, Automatic quadrilateral and hexahedral meshing of pseudo-cartesian geometries using virtual subdivision, Master's thesis, Brigham Young University (1996).

[14] T. D. Blacker, M. B. Stephenson, Paving: A new approach to automated quadrilateral mesh generation, International Journal for Numerical Methods in Engineering 32 (4) (1991) 811–847. doi:10.1002/nme.1620320410.

[15] M. L. Staten, R. A. Kerr, S. J. Owen, T. D. Blacker, M. Stupazzini, K. Shimada, Unconstrained plastering–hexahedral mesh generation via advancing-front geometry decomposition, International Journal for Numerical Methods in Engineering 81 (2) (2010) 135–171. doi:10.1002/nme.2679.

[16] R. Schneiders, A grid-based algorithm for the generation of hexahedral element meshes, Engineering with Computers 12 (3-4) (1996) 168–177. doi:10.1007/BF01198732.

[17] I. Malcevic, Automated blocking for structured CFD gridding with an application to turbomachinery secondary flows, in: 20th AIAA Computational Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, 2011. doi:10.2514/6.2011-3049.

[18] M. Zhang, J. Huang, X. Liu, H. Bao, A wave-based anisotropic quadrangulation method, ACM Trans. Graph. 29 (4) (2010) 118:1–118:8. doi:10.1145/1778765.1778855.
URL http://doi.acm.org/10.1145/1778765.1778855

[19] N. Kowalski, F. Ledoux, P. Frey, A pde based approach to multidomain partitioning and quadrilateral meshing, in: X. Jiao, J. Weill (Eds.), Proceedings of the 21st International Meshing Roundtable, Springer Berlin Heidelberg, 2013, pp. 137–154. doi:10.1007/978-3-642-33573-0_9.

[20] H. J. Fogg, C. G. Armstrong, T. T. Robinson, Automatic generation of multiblock decompositions of surfaces, International Journal for Numerical Methods in Engineering 101 (13) (2015) 965–991. doi:10.1002/nme.4825.

[21] D. Bommes, M. Campen, H.-C. Ebke, P. Alliez, L. Kobbelt, Integer-grid maps for reliable quad meshing, ACM Trans. Graph. 32 (4) (2013) 98:1–98:12. doi:10.1145/2461912.2462014.

[22] D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, D. Zorin, Quad-mesh generation and processing: A survey, Computer Graphics Forum 32 (6) (2013) 51–76. doi:10.1111/cgf.12014.

[23] W. R. Quadros, K. Ramaswami, F. B. Prinz, B. Gurumoorthy, Laytracks: a new approach to automated geometry adaptive quadrilateral mesh generation using medial axis transform, International Journal for Numerical Methods in Engineering 61 (2) (2004) 209–237. doi:10.1002/nme.1063.

[24] M. Nieser, U. Reitebuch, K. Polthier, Cubecover parameterization of 3D volumes, Computer Graphics Forum 30 (5) (2011) 1397–1406. doi:10.1111/j.1467-8659.2011.02014.x.

[25] Y. Li, Y. Liu, W. Xu, W. Wang, B. Guo, All-hex meshing using singularity-restricted field, ACM Trans. Graph. 31 (6) (2012) 177:1–177:11. doi:10.1145/2366145.2366196.

[26] G. Bunin, A continuum theory for unstructured mesh generation in two dimensions, Comput. Aided Geom. Des. 25 (1) (2008) 14–40.

[27] J. Palacios, E. Zhang, Rotational symmetry field design on surfaces, ACM Trans. Graph. 26 (3) (2007) 1. doi:10.1145/1276377.1276446.

[28] F. Kälberer, M. Nieser, K. Polthier, Quadcover - surface parameterization using branched coverings, Computer Graphics Forum 26 (3) (2007) 375–384. doi:10.1111/j.1467-8659.2007.01060.x.

[29] N. Ray, B. Vallet, W. C. Li, B. Lévy, N-symmetry direction field design, ACM Trans. Graph. 27 (2) (2008) 10:1–10:13. doi:10.1145/1356682.1356683.

[30] Y. Hon, M. Li, Y. Melnikov, Inverse source identification by green's function, Engineering Analysis with Boundary Elements 34 (4) (2010) 352 – 358. doi:10.1016/j.enganabound.2009.09.009.

[31] ITI Transcendata, CADfix website, `http://www.transcendata.com/products/cadfix` (Accessed: 02/2013).

[32] H. J. Fogg, C. G. Armstrong, T. T. Robinson, Multi-block decomposition using cross-fields, in: J. P. Moitinho de Almeida, D. P, C. Tiago, N. Pares (Eds.), Proceedings of Adaptive Modelling and Simulation, Lisbon, 2013, pp. 254 – 267.

[33] TrueGrid, TrueGrid website, `http://www.truegrid.com/QualityMesh.html` (Accessed: 06/2015).

[34] E. Ruiz-Girones, J. Sarrate Ramos, Automatic generation of quadrilateral structured meshes using linear programming and transfinite interpolation, in: 6th Workshop on Numerical Methods in Applied Science and Engineering (NMASE 07), Vall de Nuria, 2007.

[35] N. Mukherjee, An art gallery approach to submap meshing, Procedia Engineering 82 (0) (2014) 313 – 324, 23rd International Meshing Roundtable (IMR23). doi:10.1016/j.proeng.2014.10.393.